Dependability Track

Ken Birman, Cornell
Ricky Butler, NASA (Chair)
S.-K. Chin, Rome Labs
Ben Di Vito, ViGYAN
Danny Dolev, Hebrew Univ.
Michelle Hugue, Opsimath Research
Gary Koob, ONR
Carl Landwehr, NRL
David Luginbuhl, AFOSR
Keith Marzullo, UCSD
John Rushby, SRI
Rick Schlichting, UofAriz.
Dan Siewiorek, CMU
Chris Walter, WW Group

July 25, 1995

1 Introduction

The goal of this workshop was to to develop a research agenda that focuses on (1) how to achieve high assurance for each of the following four classes of system properties: real-time, security, safety, and dependability/fault-tolerance and (2) how to achieve high assurance for systems that must satisfy two or more classes of properties simultaneously.

Towards that end, the dependability track considered the following systems: air traffic control, digital avionics, enterprise computing, C^4I, Telephone systems, Satellite systems, the Information Super Highway, and Cellular Phones. We identified the fault tolerance requirements of each system, and delineated the problems that arose, especially as a result of conflicts between properties assigned to different classes. The analysis revealed a significant need for integrated methods that simultaneously deal with combinations of properties chosen from at least two different property classes. The dependability track then converged upon the view that a framework is needed to orchestrate a unified research program aimed at integrating these topics. This framework seeks to bring about a grand synthesis of the research disciplines associated with the four classes, which have largely worked separately in the past.

It should be noted that the ability to formulate this framework is a consequence of a combination of recent trends in the various areas, and is complemented by new architectures for distributed computing systems. In each of the four areas, recent technology trends favor a shift from special-purpose hardware and software solutions to solutions that embed standard off-the-shelf components. These trends will only accelerate later in the decade. Meanwhile, the enormous performance increases that have occurred in computing components are making the performance and architectural demands of the different areas seem less stringent than in the past, and less at odds with one another. Finally, there has been a trend to construct modular operating systems in which special purpose systems can be built out of general purpose components. This offers a structure within which a common framework could reside, to be specialized for each class of high assurance computing (HAC) system. There are a number of benefits achievable from pursuing this vision, as elaborated below.

In the following sections the rationale for this grand synthesis is developed. First, specific application areas where a lack of integrated solutions has resulted in significant problems are explored. Second, places where current research has not solved fundamental issues are elaborated. We then briefly discuss the need to stimulate, through increased social awareness, a public market for high assurance computing, and the need to display, through large-scale technology demonstrations of solutions to "National Challenge Problems", that technology can satisfy such a market. Finally, we propose a framework for the grand synthesis and close with other specific recommendations.

2 Example Systems Requiring Integrated Methods

Economic pressures have pushed industry to develop more integrated systems that must satisfy complex combinations of properties from several of the classes described in the introduction. However, tools and techniques that scale up to large systems have been lacking. The available tools either (1) address properties in isolation from each other, or (2) only apply to small or medium size systems. In this section, we review three specific example applications which demonstrate the need for better tools.

2.1 Air Traffic Control

The dependability track included several researchers with knowledge of the experience of IBM and Loral in undertaking to develop the AAS, a proposed next-generation air-traffic system contracted by the American FAA. We note, however, that the discussion of this topic was informal and may not represent the official views of the FAA or the development teams at IBM or Loral.

The air traffic control (ATC) system upgrade has encountered several difficulties. As originally conceived, the project was intended to take advantage of a relatively mature technology, delta-T atomic broadcast. However, this atomic broadcast theory had not been developed with the demanding real-time constraints of the ATC in mind, and had not been elaborated into a complete fault-tolerance methodology for building large, complex distributed systems. Although intuitively well matched to the ATC problem, these technologies needed to be integrated into well supported programming tools and environments, and scaled to deal with the requirements of a very ambitious development schedule.

Ultimately, it seems, this process of scaling and integration broke down. In an early stage of the effort, the fundamental protocols had to be changed because they imposed communication delays on the order of 200-300 ms for reliable message passing, which was far too long. However, when the protocols were squeezed down to give better delay characteristics, the fault-tolerance features of the model were compromised. This led to an implementation that could not meet its reliability goals without further research to restore fault-tolerance.

Similarly, a protocol for tracking system membership (a key problem in fault-tolerance) ran into trouble. This algorithm was based on a heartbeat, by which processors could detect one-another's failure by counting missed beats until a threshold was crossed. The heartbeat rate was originally too slow to detect failures with the desired real-time responsiveness. However, when the heartbeat rate was increased to enhance fault-detection, the heartbeats saturated the network, spawning yet another research challenge to be addressed immediately.

A fundamental problem of the ATC effort may have been that the design was based upon a "kernel-centric" structure which had to be modified to meet the real-time properties of the application, and onto which the distributed fault-tolerance protocols were then grafted. The original strategy of using methods that were not designed for a hard real-time arena and 100% safety margins did not work, introducing delays that contributed (with other factors) to the serious problems eventually encountered in the overall project. This was seen by many members of the dependability track as a nearly inevitable outcome, because the fundamental theoretical research was not in place to solve this kind of problem, and because the existing theoretical results were not reflected in a corresponding integrated, scalable, and commercially viable software infrastructure. Forced to build its own, theoretically sound and consistent framework, the development team was turned into a high-risk research team operating on a tight schedule.

2.2 Integrated Modular Avionics

Digital systems for commercial and military aircraft are undergoing a radical change: the fundamental architecture of these systems is moving from "federated" to "integrated". In a federated system, each aircraft function resides on its own computer. In an integrated system, there is a shared computing resource on which multiple functions execute. The issues that must be addressed when integrating previously federated applications include:

- Ensuring the isolation of applications that share common resources, such as the processor or memory, so that a faulty application cannot interfere with other applications, especially that non-critical tasks cannot interfere with critical tasks.

- Providing reliable channels for communication between applications.

- Responding to external inputs or controlling external outputs within fixed time limits.

Thus, there is a need for a *virtually absolute guarantee* of:

- Space partitioning One application must be prevented from corrupting another.

- Time partitioning A given application must not be able to prevent another application from obtaining sufficient CPU time.

Since there is no theory available with which one could build a flexible fault-tolerant, real-time system that offers time and space partitioning, industry has adopted a bus-centric table-driven static solution. While this is clearly the prudent approach, the resulting systems do not offer the anticipated advantages such as (1) simple interfaces, (2) easy modification of the applications or (3) the ability to load application software from multiple vendors.

A good example of where the theory is deficient is in real-time scheduling. Although rate-monotonic scheduling (RMS) has grown to a fairly mature state, it is still inadequate for fault-tolerant systems that are used in avionics systems. The RMS theory has not addressed the problem of scheduling redundant tasks on a multi-channel (i.e. multiple processors) fault-tolerant system in a way that provides immunity to upset from electromagnetic interference (EMI) or high-intensity radiated fields (HIRF). In particular, the execution sequences should be different so that an environmentally-induced upset will affect different tasks on the multiple channels, yet still enable a majority vote to complete before the transport-delay deadline has been reached. Also, mode changes can be triggered by asynchronous messages from other systems. The redundant channels must reach agreement on these events. Thus, there is a need for an integrated scheduling theory that addresses the fault-tolerance and real-time properties simultaneously.

Given such a theory, of course, the same issues raised in the ATC example would become relevant. The theory would need to yield practical solutions that operate on commercial, off-the-shelf components (COTS) and, ideally, that could interoperate with commercial application software for developing software, for managing the hardware platform, and for performing other non-avionics tasks. Thus a pattern similiar to that of the previous case emerges:

a need for basic research, as well as a need to embed those results into a more useful broad framework.

2.2.1 Space Partitioning Problem Can Draw From Security Work

The need to guarantee the isolation of different criticality tasks that share the same memory has much in common with the “security kernel” concept (or, rather, its more primitive foundation, the “separation kernel”) used in computer security. The criticality level of the kernel operating system must be at least that of the highest task. The kernel OS is an ideal candidate for the most rigorous formal methods available. Much work has been performed to enable the formal verification of security kernels, but little of this work has been applied successfully in COTS-based environments, creating a tension between the goal of multi-level system security and the pragmatic requirement that systems be constructed from inexpensive commodity components and with standard development tools.

The connection between dependability and security was identified as a dual one. As observed, security and dependability share a need for space partitioning or “isolation” technologies. At the same time, security systems are threatened by failure, which can result in denial of service attacks. Existing security technologies revolve around non-replicated authentication authorities, creating problems as these systems are scaled: the probability of being partitioned away from the authentication subsystem rises to the degree that one desires continued operation despite such failures. Yet the principle of replication or decentralization of authentication also creates a security exposure.

The development of technologies combining security and dependability (high availability) was thus identified as a priority. Goals of such an integration would be to identify a common set of mechanisms for component isolation and fault-isolation that might be useful both for space partitioning in embedded applications and for security purposes; and to better understand the issues associated with avoiding denial of service caused by failures in a secure distributed application.

2.3 C^4I

The development of the Command, Control, Communication, Computer and Intelligence (C^4I) systems for the Dept. of Defense has been based upon technology developed for the commercial market. In particular, the remote procedure call (RPC) technology has been used extensively. Unfortunately, the RPC end-to-end approach does not work well here. It prevents fault-tolerance at the communications layer because RPC failure reporting is triggered by timeouts, which may result in erroneous or inconsistent failure detections. Fault-tolerance requires replication, which is costly to implement using an RPC substrate. Moreover, commercial RPC methods (and other distributed computing technologies) are based upon an assumption that a processor failure will cause the processor to halt. The resulting systems are not robust in the presence of faults that fall even slightly outside of this model, much less to true Byzantine failures.

This example exposes what may appear to be a fundamental conflict between the needs of security and fault-tolerance here. The monitoring techniques used to detect and tolerate failures need global data, whereas the security requirements of a C^4I application necessitate a restriction on information flow (i.e. to prevent covert channels). Also the security concept is widened here. While most fault-tolerant systems assume a basically benign world of well-tested components subject to a low residual bug rate and hardware or communication failures, C^4I systems confront a more hostile world model. These systems must be protected from hackers/terrorists that seek to disrupt the availability of the system rather than to just steal information.

Methods and standards are needed that address both security and fault tolerance in an integrated way. In particular, an ISO-like hierarchy of abstractions that provides a spectrum of these properties from minimal capability to a full-spectrum of capabilities, combined with a security model, could be very useful.

3 Fundamental Research Deficiencies

These summaries are typical of the discussion and review undertaken by the dependability track, jointly with other tracks. Our analysis led to a multi-level view of the issues confronting the HAC community, at least in regard to dependability.

First, we identified a broad societal requirement to stimulate greater concern for and attention to robustness of the commercial infrastructure. Viewed in this light, the current atmosphere of crisis surrounding security violations of the Internet, and the difficulties encountered by the ATC effort and similar highly publicized large-scale systems engineering projects (the Denver Airport comes to mind) may be positive factors, that will help motivate the major industrial players to accommodate more readily assured technologies in their basic products and communication service offerings. Lacking such an industry buy-in, a special-purpose infrastructure meeting the special needs of the HAC community might still fail to meet the broader need, which is a technology base that can interoperate with COTS technologies.

Second, there is a significant need for further study and practical research of integrated assurance technologies. Across the board, we found a pattern of isolated results that are known to work in the laboratory or in small-scale experiments, but are not available in a form that can be reused by others or applied to large-scale problems. To break this barrier, several steps are needed. First, it will be necessary to develop an engineering science of large-scale systems design buttressed by sound theory and the sorts of design tools that are widely used today in less assured forms of computing. Second, large-scale demonstrations are needed ("national challenge" projects) that could develop practical methods of scaling these fundamental ideas from the laboratory into practice. From these, collatoral materials are needed that would enable practitioners to learn from the demonstrations and make direct use of the reusable technologies and the reusable architecture.

This leads to our third requirement, a unified architectural framework within which both the theory and practice of highly assured computing, in its varied forms, could be integrated and interplay. Such an architectural framework would have a descriptive aspect and a practical aspect. On the former (descriptive) side, the architecture should let us characterize the properties of a system in a rigorous way, permitting complex reliability properties to be described and verified using modular representations of the system and compositional verification techniques. On the latter (practical) side, a concrete software environment is needed that could realize this architecture over commercial-off-the-shelf (COTS) components, so that applications requiring highly assured technologies could exploit the architecture in a practical, portable manner. The framework should be sufficiently general to model existing systems, admitting the possibility that some layers may be realized in hardware or integrated with the application, or the possibility that some layers may have been merged in the interest of optimization. This would allow existing systems to be decomposed and analyzed within the framework and support reengineering. It would also avoid the potential appearance that one approach was being imposed on the community. We elaborate upon this idea later, but it basically leads to a view that a standard framework for building highly assured computing systems is perhaps within reach in 1995, and would be of broad benefit to the entire field.

Finally, our review identified a number of specific topics in need of future research: (1) a real-time atomic broadcast primitive that is fast, (2) a practical way to use such a primitive to build fault-tolerant systems, (3) a space/time partitioning technology for building critical embedded systems, (4) a realization of RMS scheduling in a fault-tolerant architecture, (5) methods for protecting against denial of service in secure systems, and (6) formal methods to assure that all of the above are correct and safe. These are enumerated in a later section.

4 Industrial Adoption

The dependability track devoted little time to the discussion of issues associated with the industrial adoption of HAC technologies. While we feel there is a manifest societal need for HAC technology in general, and dependability in particular, we also recognize that the economic incentive to implement it, especially on the part of commercial vendors, grows slowly. Industry is historically driven by primary factors such as cost, performance, and time to market. Secondary factors such as dependability are accorded lower priority, much lower, we feel, than is warranted by the attendant risk. This issue is clearly in need of further discussion and study by Federal policy makers. A strong case exists that the government has a clear interest in promoting greater dependability in the computing and communications infrastructure.

Shortly after the HAC workshop met, a British bank failed when a single individual, in violation of bank policy, placed a large financial bet and lost. Post-disaster analysis of this incident indicated that the performance of information systems used to make high-stakes financial transactions has benefitted greatly from advances in technology, while insufficient attention has been paid to implementing safeguards that prevent error and abuse. It is chilling to think that even had the bank wished to insert such safeguards, the technology base needed to prevent this disaster is not yet at hand.

At heart, what is lacking is a public, and even professional, awareness of the risks posed by low-assurance computing. While increasing such awareness is essential, it is not a research problem per se. The government has a clear role to play in other venues to help increase public and industrial awareness. We simply note here that there are precedents for such a shift. Consider the automotive industry, where passenger safety is now a major factor in product design and customer buying decisions. Twenty years ago auto executives bitterly fought the introduction of airbags into their cars. Now those same executives go on television to brag about how many airbags they place in their vehicles.

HAC technologies are the airbags of the computing industry. Useful safety technologies, such as security and authentication schemes and software fault-tolerance solutions exist today, and are available for application by industry. But these promising first steps are as yet inadequate to address the needs of an information infrastructure that is growing explosively. It is essential to advance the technology now so that as industrial and public awareness of risk continues to grow, there will be adequate technology available to meet the demands of a increasingly information-dependent society.

A principal block to successful transition of theoretical results into real products is the relative inaccessability of the information needed to make the theory practical or to identify candidate theory that matches the problem. For, unless one is instantaneously handed a black box solution to a given problem, information must be transferred as well as technology. Any parties involved in the transfer must negotiate a common frame of reference (or certainly attempt to) in which they are both talking the same language, so to speak. Typically, much practical real-world information is either unavailable or of little interest to academics or theorists because they are trying to develop the big picture, the meta-meta-object, that will generate instantiations with specified characteristics. Furthermore, the "real data" needed to improve the accuracy of model assumptions such as failure rates, fault arrival rates, repair rates, associated with reliability and availability modeling, are often not even recorded. When such information is recorded, it rarely moves outside of the realm of proprietary technology. Similarly those attempting to solve the "real problems in the field" only rarely have the opportunity or desire to enter the world of the theorists. The window of commercial opportunity for a potential product is often significantly shorter than the time needed to research and understand any models or formalisms underlying critical attributes of the services to be provided. Also there is often an insufficient understanding of the problem at hand which hamper's one's ability to identify any theory that applies. The opposite of this situation is also true. There are situations where theory has been developed but no one knows how to implement it with currently available technology or the cost of adopting the new technology is prohibitive.

Clearly, more information characterizing the different application classes that we identified must be made available to researchers. Since industry is understandably loathe to risk corporate secrets, something new must be done to facilitate the information gathering process. One suggestion has been to have academics spend a year working in a certain industry. Alternatively, companies could be encouraged (through partial funding) to sponsor a post-doc type program, or more coop programs in which researchers can get some idea of the complexity and scope or real-world problems. Note that assessments of the behavior of fielded systems in their target environments are also needed to accumulate a sufficient knowledge base to identify potential problem areas for new systems.

5 National Challenge Problems

Similarly to the social issue, the dependability track devoted little time to the discussion of suitable projects for large-scale technology demonstrations, or the attendant practical issues. This need was repeatedly identified in meetings with other tracks, and resulted in a number of suggestions that were not pursued in detail. However, the track does wish to stress that laboratory demonstrations of solutions can no longer be viewed as successes in and of themselves. The broader issues of integration and of achieving high degrees of assurance in large, complex systems were prominent in all areas that were reviewed and discussed. Only large-scale technology demonstrations can bridge the wide gap between theory and practice in the disciplines we represent.

6 Framework For the Grand Synthesis

We propose that an integrated architecture be developed, enabling one to compose a system out of reusable blocks having various combinations of fault-tolerance, real-time, safety, and security properties. This architecture would be ISO-like in that it would be layered and hierarchical, but unlike ISO would not be based upon end-to-end protocols.

For example, many dependability techniques for distributed systems are based on some form of cooperation within groups of components. NMR replication and voting lie at the heart of the many critical embedded systems in widespread use today. Duplication of data is used for load-balancing and primary/backup computing. Also, replication is used for fault-tolerance in commercial hardware design. Thus, in contrast to an ISO protocol layering, which focuses on connections between pairs of processes, we believe that a hierarchical layering based upon groups of cooperating processes could represent a fruitful direction for study. In the case where a group has only a single member, this collapses to a well-known layering such as is seen in ISO. Unlike ISO, however, a group model can potentially describe a great number of fault-tolerance solutions spanning many basic approaches, and is conducive to a simple software architecture similar to the popular microkernel approach for building operating systems.

This layering approach has been successfully used to structure fault-tolerant distributed software systems, such as Horus, Transis, xKernel (Psync) and Totem. In each of these cases, the specific capabilities desired in a specific computing setting are achieved by building highly layered software, with each layer refining and extending the properties of the layer below it. AT&T's Rampart system uses the same approach to deal with byzantine failures in a security environment, and has reported similar success. Although non-trivial, it would appear that such an architecture could be extended to also describe the properties of embedded systems for fault-tolerant control, and to be compatible with emerging techniques for dealing with real-time requirements in complex environments. Such a layering approach would then be capable of describing multiple schemes for highly assured computing in a single framework.

In addition, this layering approach could be extended to include legacy software which was not originally designed to HAC standards. Legacy software will remain the dominant form of software for the near term future.

Layering allows the definition of fault confinement regions between HAC modules and legacy software. Suitable error detection and error handling capabilities can be defined at these boundaries to extend HAC attributes to the legacy software. Inclusion of legacy software into a dependable system has been demonstrated for both the Mach/Unix environment as well as DOS/Windows environments.

Conceptually, such a layered approach can be characterized as a Lego$^{\text{tm}}$ building-block treatment of highly assured computing systems. For example, one type of block might overcome communication failures, another might enforce authentication on the communication connections between components. A more complex building block could run a TMR voting protocol on inputs supplied by the members of a group, provide clock synchronization, or implement the virtual synchrony properties on which many fault-tolerant distributed systems are based. In concept, these building blocks should be arbitrarily stackable: one should be able to omit blocks, stack them multiple times, or reorder them. In practice, of course, some blocks will only be useful in specific settings, while others are of more general utility. Specific stacks supporting the most important styles of dependable computing would be needed, at a minimum, to establish the viability of the approach as a general standard.[1]

To be useful, a framework must correspond to one or more flexible, efficient implementations. The success of ISO is clearly tied to its utility in describing both the OSI protocols and also the TCP/IP protocol suite. Similarly, the framework we seek should be supported by a technology base at least capable of demonstrating separate solutions to dependability problems in software fault-tolerance, synchronous fault-tolerance for embedded components, security, and distributed real-time communication. Ideally, the software base should go further and address the need to integrate more than one of these properties in a single application, but we view this as a question from which a number of research topics can be derived. The advantage of doing so in the context of a standard architecture and framework is that the results can then be reused in existing systems compatible with the framework,

[1]For example, a single system could support the ability to form groups of components that communicate over a network, with communication properties entirely determined by the application. A specific use of groups would correspond to a specific stack of Lego blocks. The individual blocks would implement microprotocols, corresponding to a modular decomposition of the algorithm supporting a complex dependability property.

something that is not often the case today.

The needed integrated architecture should have the property of *composability:* it should be based on building block components that have uniform interfaces. Adding a new block should not destroy the properties of the other blocks in the system, but rather should extend those properties in a predictable way. The critical algorithms used in these building blocks should be formally verifiable, and tools supporting verification and testing should be developed for the architecture. Much as the security community has pressed for trusted versions of the Mach operating system, highly assured versions of the architecture itself should be a goal of this standardization effort.

7 Other Specific Recommendations

This section enumerates a number of additional recommendations that emerged from discussions with the dependability track. While lacking the broad visionary nature of our recommendations for a standard architecture, these are important research directions and we strongly encourage the HAC research program to address them.

1. Development of formal analysis tools and methods to verify the fundamental algorithms of the integrated architecture.

2. Effective methods for re-engineering existing (inadequate) database designs.

3. Standards for building object-oriented database systems.

4. Development of certifiable methods for time and space partitioning on integrated systems.

5. Better integration of transactional and non-transactional software tools.

6. Formalization of preemptive scheduling algorithms.

7. 4GL development tools.

8. Techniques for handling multiple simultaneous faults (e.g. from HIRF).

9. System design principles that permit the use of COTS without (a) undue risk of unwanted disclosure of sensitive information, or (b) undue risk of loss of operational capability to flawed or sabotaged hardware/software components.

10. Customizable transaction properties: single system in which the properties for a particular operation are configurable based on need, and in which the application pays only for properties it uses.

11. More rigor in requirements analysis and other early lifecycle activities.

12. Use of analysis to reduce testing cost.

13. For critical availability, techniques needed to overcome potential correlated outages caused by failures of the transactional technology itself.

14. Development of certifiable methods for customizing protocols.

15. Flow control in complex distributed systems.

16. Better embeddings of technology into OS platform.

17. Standards for basic process-group interfaces.

18. Demonstrations of scalability to very large environments.

19. Research needed on partition tolerance, high-level technologies for building applications that remain available during partitioned and disconnected operation.

20. For critical availability, techniques needed to overcome potential correlated outages caused by failures of the process group technology itself.

21. Major improvements in system management techniques.

22. Effective methods for presenting certifiers and accreditors sound technical information they can use to make rational choices.

23. Effective methods for integrating existing, separately developed systems so that they can evolve toward the integrated system described here.